UNIVERSIDADE DO VALE DO TAQUARI
CURSO DE ENGENHARIA DA COMPUTAÇÃO

# AUTHTOKEN: AUTENTICAÇÃO DE ACESSOS VIA QRCODE PARA VALIDAÇÃO DE PRESENÇA UTILIZANDO DISPOSITIVOS DIGITAIS

Diego Gabriel Bressler

Lajeado, 16 de novembro de 2020

Diego Gabriel Bressler

# AUTHTOKEN: AUTENTICAÇÃO DE ACESSOS VIA QR CODE PARA VALIDAÇÃO DE PRESENÇA UTILIZANDO DISPOSITIVOS DIGITAIS

Projeto de Monografia apresentado na disciplina de Trabalho de Conclusão de Curso II, do curso de Engenharia da Computação, da Universidade do Vale do Taquari - Univates, como parte da exigência para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Luis Antônio Schneiders.

Lajeado, 16 de novembro de 2020

# RESUMO

O avanço tecnológico está trazendo inúmeros benefícios e recursos, que podem auxiliar no cotidiano das pessoas. Atualmente existem diversas formas para controlar o acesso a eventos ou lugares que utilizam sistemas de monitoramento, entretanto, vários destes sistemas também realizam o controle de acesso por meio do uso de dispositivos embarcados, que realizam a comunicação entre o *hardware* e o *software*. O presente trabalho busca desenvolver um protótipo para auxiliar na validação dos acessos em eventos ou lugares privados, de modo a autenticar, autorizar e manter registro destes acessos. Para desenvolvimento deste trabalho é utilizada a tecnologia *QR Code*, responsável por identificar os acessos por meio de uma câmera que é conectada a um microcontrolador *Raspberry Pi,* que realiza a leitura dos *QR Code* que valida a permissão para o acesso. Trata-se de um trabalho em caráter experimental, executado a partir da configuração de um protótipo, a partir de dispositivos convencionais. Os resultados alcançados são promissores e validam a hipótese de que é possível desenvolver sistemas de controle de acesso funcionais e de baixo custo, a partir de dispositivos móveis.

**Palavras-chaves:** *QR Code, Raspberry Pi*, Autenticação.

# ABSTRACT

Technological advancement is bringing countless benefits and resources, which can help in people's daily lives. Currently, there are several ways to control access to events or places that use monitoring systems, however, several of these systems also perform access control through the use of embedded devices, which perform the communication between hardware and software. The present work seeks to develop a prototype to assist in the validation of accesses at events or private places, in order to authenticate, authorize and keep a record of these accesses. To develop this work, the QR Code technology is used, responsible for identifying the accesses through a camera that is connected to a Raspberry Pi microcontroller, which performs the reading of the QR Code that validates the permission for access. It is an experimental work, performed from the configuration of a prototype, from conventional devices. The results achieved are promising and validate the hypothesis that it is possible to develop functional and low-cost access control systems from mobile devices.



**Keywords:** QR Code, Raspberry Pi, Authentication.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| API | Application Programming Interface |
| CLI | Command-Line Interface |
| CPU | Central Process Unit |
| CSS | Cascading Style Sheets |
| GB | GigaByte |
| HDMI | High-Definition Multimedia Interface |
| HTML | Hypertext Markup Language |
| HTML5 | Hypertext Markup Language version 5 |
| HTTP | Hypertext Transfer Protocol |
| ID | Identity |
| IdM | Identity Management |
| IEC | International Electrotechnical |
| IOS | iPhone Operating System |
| ISO | International Organization for Standardization |
| JSON | JavaScript Object Notation |
| MHz | Mega-Hertz |
| NBR | Normas Brasileiras |
| NFC | Near Field Communication |
| PIN | Personal Identification Number |

| | |
|---|---|
| QR Code | Quick Responsive Code |
| RAM | Random Access Memory |
| REST | Representational State Transfer |
| RFID | Radio Frequency Identification |
| SOAP | Simple Object Access Protocol |
| SoC | Security Operations Center |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| USB | Universal Serial Bus |
| WLAN | Wireless Local Area Network |
| XML | eXtensible Markup Language |

# SUMÁRIO

# 1 INTRODUÇÃO

O uso crescente da tecnologia vem trazendo inúmeras formas para a realização de autenticação, seja para desbloquear um *smartphone*, validar acesso a um evento, acessar um local privado, ou um sistema que contém informações importantes. Cada uma destas maneiras de autenticação tem como objetivo tornar as informações e locais mais seguros.

Com todo avanço tecnológico dos últimos anos, os dispositivos móveis tornaram-se indispensáveis e acessíveis. Conforme a Pesquisa Anual de Administração e Uso de Tecnologia nas Empresas, realizada pela Fundação Getúlio Vargas de São Paulo (FGV-SP), divulgada em abril de 2019, o Brasil tem cerca de 230 milhões de celulares ativos no país, um aumento de 10 milhões ao comparar com o ano anterior, ou seja, existe, em números absolutos, pelo menos um dispositivo móvel para cada pessoa.

Porém, segundo Meirelles (2019), responsável pela pesquisa, a venda destes dispositivos tende a diminuir, pelo fato de não se precisar de mais de um aparelho por pessoa, pois hoje um *smartphone* está suficientemente robusto, englobando várias funcionalidades em um mesmo aparelho.

Desse modo, considerando o avanço das tecnologias, muitos problemas são resolvidos através da utilização de *smartphones*. Umas destas novas tecnologias presentes nos *smartphones* é o *Quick Response Code (QR Code)*, que nada mais é do que um código bidimensional que contém informações que podem ser captadas por uma câmera incluída nos dispositivos móveis. Atualmente, ele é conhecido por

ser utilizado para realizar pagamentos de contas, para o acesso de sites, entre outros.

O *QR Code,* entretanto*,* pode ir muito além disso, podendo ser utilizado para validar a presença em um evento, pode substituir uma passagem aérea, disponibilizar *link* de acesso a um cupom eletrônico, cupons de desconto, entre outros (COELHO, 2013). Pensando nisso, o *QR Code* pode ser utilizado em diversos ramos de atividade, incluindo o ramo educacional.

No ramo educacional é possível aplicar serviços suportados por esta tecnologia como o controle de acessos a eventos, cursos e locais como bibliotecas, laboratórios e salas de aula. Esta utilização pode agilizar a entrada das pessoas, além de poder monitorar, autenticar, manter registros e controlar o acesso a estes locais.

Partindo deste pressuposto, torna-se necessária uma solução para o gerenciamento e controle dos acessos a um evento ou local e que esse ofereça praticidade ao permitir ou negar algum acesso, assim como segurança no monitoramento, identificação, autorização e armazenamento dos registros utilizando dispositivos digitais.

Para atender aos requisitos citados acima, é desenvolvido a solução *AuthToken* para dispositivos móveis onde o usuário irá acessar um *QR Code* através do seu *smartphone*, para então acessar locais específicos, passando por um dispositivo digital que irá fazer a leitura do *QR Code* recebido, validando e registrando o acesso ao evento ou local .

## 1.1 Objetivos

O objetivo do presente trabalho é o desenvolvimento de um dispositivo que visa otimizar o processo de autenticação e autorização com base em *QR Code* e que possa, além de gerenciar o processo de autenticação e autorização, fornecer aos administradores uma consulta dos registros de acessos.

## 1.2 Objetivos Específicos

São objetivos específicos desta pesquisa:

- Construir um referencial teórico sobre métodos de autenticação e sobre a tecnologia *QR Code*.
- Analisar trabalhos relacionados com tecnologias similares ao *QR Code* e métodos de autenticação.
- Montar o protótipo para a leitura e envio de dados para a aplicação.
- Desenvolver um aplicativo para recebimento dos *QR Codes* pelos usuários.
- Realizar a validação dos dados lidos pelo protótipo.
- Validar a solução desenvolvida em laboratório.
- Analisar a viabilidade do uso real do protótipo desenvolvido.

## 1.3 Justificativa

Este trabalho busca facilitar o controle e gerenciamento do acesso a eventos ou locais específicos. Além disto, pretende-se oferecer ao controlador desta aplicação uma ferramenta de monitoramento, disponibilizando formas de gerenciar com maior segurança a liberação do acesso a determinados eventos ou locais.

## 1.4 Estrutura do trabalho

Este presente trabalho foi estruturado em seis capítulos.

No Capítulo 1 discute-se o tema abordado neste trabalho, considerando o cenário atual do tema apresentado, o problema a ser abordado e os principais objetivos do mesmo.

No Capítulo 2 é realizada uma revisão bibliográfica para fundamentação e compreensão do tema apresentado, além de contextualizar as principais ferramentas que integram a solução apresentada.

No Capítulo 3 são apresentados os trabalhos relacionados e a abordagem da utilização de ferramentas similares, portanto relevantes ao presente trabalho.

O Capítulo 4 apresenta os métodos e materiais que a presente a pesquisa, assim como as tecnologias que sustentam a elaboração do protótipo.

O Capítulo 5 apresenta o desenvolvimento do protótipo e também os testes de funcionalidade e acuracidade.

Por fim, no Capítulo 6 são apresentadas as observações e considerações finais.

# 2 REVISÃO TEÓRICA

Esta é a parte do trabalho que apresenta brevemente uma revisão dos principais fatores, como fontes, obras, referências que tratam do tema em questão. O objetivo da revisão teórica é destacar, resumir ideias já formuladas por outros autores, realizando comparações.

## 2.1 Sistemas de controle de acesso

Na definição de Marcondes (2020), os sistemas de controle de acesso, são mecanismos que interligam *software, hardware* e suporte humano, com o objetivo de estabelecer o acesso a um determinado local, para aqueles usuários que estejam previamente cadastrados e são corretamente autenticados, seja por cartões, senhas ou até mesmos traços biométricos.

Marcondes (2020), complementa que as permissões de acessos, concedidas ou não, são registradas em um servidor apropriado e que, além disso, são gravados os registros de entrada e saída de um local ou sistema. Deste modo, a característica fundamental destes sistemas é garantir que os envolvidos sejam pessoas, objetos ou informações, estejam seguros e em um ambiente totalmente monitorado (MARCONDES, 2020).

Para Marcondes (2020), estes sistemas devem possuir outras características importantes, tais como: não serem atrativos para a tentativa de acessos não permitidos. Permitir aos administradores do sistema um amplo controle, seja para saber quem está acessando cada local, alertar quando há uma tentativa de acesso

negada, estar preparado para responder quando há tentativas de infringir o acesso, neutralizando as tentativas de acessos indevidos, além de relatórios destes acessos.

De acordo com Marcondes (2020), comenta que o funcionamento destes sistemas, nada mais é que o monitoramento de entrada/acessos e saídas de um local ou sistema protegido, no qual utilizam de *hardwares* de bloqueios como: catracas, dispositivos identificadores (*Radio Frequency Identification* (RFID[1]), de cartão, senhas, biometria, entre outros).

Estes *hardwares* se comunicam com um controlador que irá mandar as solicitações de acesso para um servidor, responsável por realizar a validação do acesso. Sendo válido, o controlador faz a liberação do dispositivo que está efetuando o controle do acesso (MARCONDES, 2020).

Marcondes (2020), destaca que estes sistemas, devem possuir um *software* robusto para seu gerenciamento e deve permitir ao administrador, o cadastro das permissões de acesso de cada usuário, devem manter todos registros e tentativas dos mesmos salvos, monitoramento em tempo real dos acessos e relatórios para análises.

## 2.2 Métodos de autenticação

Os métodos de autenticação são ferramentas utilizadas para averiguar a identidade de um indivíduo, visando identificar quem é e se possui permissão para acessar um ambiente. Antunes (2014), resume os métodos de autenticação em mecanismos tecnológicos que buscam examinar e comprovar a identidade de uma pessoa e comprovar quem ela alega ser.

Esses métodos são divididos em três conceitos de autenticação, sendo eles de conhecimento, de propriedade e de características. Estes métodos de autenticação ainda podem ser combinados, esta junção dos conceitos é denominada de modelo em dois fatores.

[1] O RFID é uma etiqueta que troca informações por radiofrequência.

### 2.2.1 Autenticação baseado em conhecimento

De acordo com Antunes (2014), este método é o mais conhecido, e fácil de ser usado, pois só é necessário que o agente que realizará a autenticação conheça os dados tais como nome, senha, *Personal Identification Number* (PIN), entre outros. Apesar de ser de fácil uso, caso o agente não contenha uma senha forte, existe um grande risco que é a facilidade de alguém conhecer essa senha, pois pode ficar tentando até descobri-la. Por outro lado, se esta senha for classificada como senha forte, este risco é reduzido.

### 2.2.2 Autenticação baseado na propriedade

Na definição de Antunes (2014), neste método é necessário que o usuário possua algum dispositivo para se autenticar, assim como cartões ou *tokens*. O *token* é um recurso que gera uma nova senha a cada determinado período que, combinado a outra técnica de autenticação, garante a legitimidade do acesso.

Para Antunes (2014), estes dispositivos tornam este método mais seguro que o do conhecimento, além de ser mais rápido ao usuário, que não precisa lembrar e informar uma senha, que normalmente para ser segura é extensa.

Antunes (2014) ressalta que este método ainda contém alguns riscos, como a clonagem de cartões, ou casos em que a empresa que gera os *tokens* é invadida, fazendo com que o invasor se passe pelas vítimas.

### 2.2.3 Autenticação baseado em característica

Para Antunes (2014), este método é o mais rígido na questão de autenticação, pois na teoria, somente a própria pessoa pode se autenticar. Neste método são usadas características físicas dos usuários, como biométrica, íris, face, reconhecimento de voz, entre outros.

Apesar de serem mais seguros que os modelos anteriores, ainda contém riscos de invasores, a escolha por um leitor de qualidade também interfere, mas

como já foi noticiado, existem formas de clonar uma digital por exemplo, fazendo molde de silicone (ANTUNES,2014).

### 2.2.4 Autenticação em dois fatores

De acordo com Donohue (2014), neste modelo é oferecido ao usuário além da forma tradicional de autenticação, uma segunda forma para garantir maior segurança, normalmente é utilizado em sites de serviços *online*, como sites de vendas, redes sociais entre outras.

> A primeira forma – em geral – é a sua senha. O segundo fator pode ser qualquer coisa, dependendo do serviço. O mais comum dos casos, é um SMS ou um código que é enviado para um *e-mail*. A teoria geral por trás de dois fatores é que para efetuar *login*, você deve saber e possuir algo a mais. (DONOHUE, 2014).

Para Donohue (2014), este método torna mais segura a autenticação, visto que um invasor, além de descobrir a senha do usuário, vai precisar descobrir o código enviado por SMS ou *e-mail*. Outro ponto positivo na utilização deste método, é poder saber quando alguém consegue descobrir a senha, pois irá receber um código para completar o acesso.

## 2.3 *Web service*

Para Ferreira (2017), *Web Services* são aplicações que atuam na rede, de forma geral, na *Web*. Essas aplicações são responsáveis pela ligação com outros serviços, no qual realizam a troca de informações em formato *eXtensible Markup Language* (*XML)* ou *JavaScript Object Notation (JSON)* padronizados, de forma ágil. Estes *Web Services* são independentes da aplicação, ou seja, podem ser feitas em qualquer plataforma ou linguagem.

Soares (2019), cita que os *Web Services* são o englobamento de diversas tecnologias, que são normalmente utilizadas no desenvolvimento de aplicações visando proporcionar agilidade nas requisições. Estes podem ser construídos em dois padrões, o *Simple Object Access Protocol (SOAP)* e o *Representational State Transfer (REST)*.

De acordo com Soares (2019), o *SOAP* é uma arquitetura orientada a serviços que pode utilizar de outros serviços para oferecer outro determinado serviço. E o *REST* é uma arquitetura que segue uma série de princípios e restrições para o desenvolvimento das aplicações.

Soares (2019), ainda complementa que *REST* é um padrão de práticas que visa determinar a forma na qual padrões como *Hypertext Transfer Protocol (HTTP*) e *Uniform Resource Identifier* (*URI*) têm de seguir para a criação de um projeto com *interface* de comunicação definida.

De acordo com Soares (2019), o uso do *HTTP* realiza a transferência de recursos através da internet, onde os métodos do *HTTP* interagem sobre estes recursos, criando uma interface única podendo ser acessada por inúmeros dispositivos. Este protocolo realiza requisições (*request*) pelo lado do cliente, e as respostas (*response*) ocorrem no lado do servidor.

Ferreira (2017), aponta que atualmente este protocolo possui diversos métodos como *GET, POST, PUT, DELETE*, entre outros. Estes métodos são usados pelo protocolo *HTTP* para executar ações sobre os recursos que podem ser executados através da *URI*. O Quadro 1, demonstra a função dos principais métodos.

Quadro 1: Métodos *HTTP*.

| | |
|---|---|
| *GET* | Solicita um recurso |
| *POST* | Cria um recurso |
| *PUT* | Atualiza o recurso |
| *DELETE* | Deleta um recurso |

Fonte: Adaptado pelo autor baseado em Ferreira (2017).

### 2.4 *QR Code*

O *QR Code* surge em 1994, apresentado em um projeto desenvolvido por Masahiro Hara, funcionário da empresa Denso Wave, uma subsidiária da Toyota, com o objetivo de identificar as peças produzidas pela empresa, visto que os

códigos de barras utilizados na época, suportavam 20 caracteres. Além disto, os códigos de barras permitiam a leitura somente por um ângulo, já o QR Code foi projetado para ser escaneado a partir de qualquer ângulo, 360º (bidirecional) (FREITAS, 2017).

Este projeto teve duração de aproximadamente um ano e meio, tendo como objetivo ter um símbolo que torna fácil a busca dos equipamentos da empresa, onde maior capacidade de informações e após feita sua leitura a resposta é instantânea, sendo superior aos códigos de barras (FREITAS, 2017).

De acordo com Freitas (2017), a empresa Denso Wave liberou a divulgação do código fonte, tornando-o livre para que qualquer empresa ou pessoa pudesse utilizá-lo, sem custo. Entretanto em 1997, é aprovada uma lei pela *Automatic Identification Manufacturer*, na qual exige que estes códigos seguissem um conjunto de regras para sua utilização, com a finalidade de tornar os códigos padronizados.

No ano de 2000, foi aprovado pela organização *International Organization Standardization (ISO)*, como um dos padrões internacionais. Dois anos depois, surgem então os *smartphones* que são capazes de fazer a leitura destes códigos, e assim impulsionando a utilização dos mesmos, pelo simples fato de tornar o acesso a sites e qualquer informação que esteja online, de forma rápida (FREITAS, 2017).

Atualmente o *QR Code* é o código bidimensional mais famoso do mundo, entretanto existem milhares de alternativas similares, sendo em torno de 100 derivados do *QR Code*, independente disso, estes códigos surgem para armazenar maior quantidade de dados que os códigos de barras mais tradicionais (KARASINSKI, 2013).

Apesar de existir há mais de 25 anos, encontra-se pouco material desenvolvido sobre o assunto, mesmo assim neste tópico, procurou-se abordar um pouco da história, o que é o *QR Code*, seus formatos e suas formas de leitura e utilização.

### 2.4.1 *QR Code* e sua estrutura

Na definição de Freitas (2017), o *QR Code* é um código de duas dimensões, composto por módulos de cor preta em um fundo branco, são semelhantes ao código de barras tradicional, mas com maior quantidade para armazenamento de informações, e possuem uma diferença na sua forma gráfica, como podemos observar na Figura 1, a diferença gráfica destes códigos.

Figura 1: Código de barras versus *QR Code*.


Fonte: Adaptado pelo autor, baseado em Karasinski (2013).

A leitura destes códigos ocorre digitalmente, é feita a partir de um *smartphone* ou *scanner*, no qual realiza a captura da imagem, e assim um programa específico decodifica as informações que contém no código lido (KARASINSKI, 2013).

Para Karasinski (2013), cada módulo deste código, tem uma função específica, iniciando pelos quadrados maiores nas três extremidades de cada código, que servem para orientação do leitor, informando onde estão os demais dados, além disso auxilia para que possa ser lido de qualquer ângulo. Existe um quarto quadrado, um pouco menor que estes três, e que fica mais ao meio do código, que tem por função alinhamento e mostrar como o código deve ser processado.

Braga (2009), complementa que do mesmo modo que os códigos de barras tradicionais, usado em boletos, precisa comprovar a integridade dos dados a serem

lidos, com intuito de saber se os *bits* em cada bloco estão corretos, para isso, existem módulos de detecção de erros.

Estes códigos possuem diversos formatos, e para saber de qual se trata, existem alguns módulos para representar isso, além disso, outra informação contida nesses códigos é o *timing*, que significa a velocidade recomendada para a leitura. Por fim, ao lado de dois dos maiores módulos, alguns destes representam a versão do código. A Figura 2 ilustra onde estão localizados cada um destes módulos citados.

Figura 2: Estrutura do *QR Code*.


Fonte: Adaptado pelo autor, conforme Jesus (2018)

### 2.4.2 Formatos

Inicialmente os *QR Code* na sua versão 1, continham 21 x 21 módulos, e armazenavam cerca de 35 caracteres números e 21 alfanuméricos, conforme a *ISO IEC* 18004, e são utilizados até hoje. Atualmente essas versões vão até a de número 40, que contém 177 x 177 módulos, que armazenam 7.089 caracteres numéricos e 4.296 alfanuméricos (Norma *ISO/IEC* 18004, 2015). A Figura 3 demonstra algumas dessas versões.

Figura 3: Exemplos de versões do *QR Code*.


Fonte: Braga (2009).

Conforme a Norma *ISO IEC* 18004, os módulos responsáveis pela detecção e correção de erros, que é responsável por garantir a integridade dos dados de cada código, estes podem ser classificadas em quatro diferentes níveis com diferentes porcentagens de capacidade de recuperação, como podemos ver a seguir (Norma *ISO/IEC* 18004, 2015):

- Nível L: 7% de capacidade de recuperação.
- Nível M: 15% de capacidade de recuperação.
- Nível Q: 25% de capacidade de recuperação.
- Nível H: 30% de capacidade de recuperação.

De acordo com Jesus (2018), estes módulos são capazes de permitir a leitura destes códigos sem a perda de nenhuma informação, mesmo que estejam danificados ou manchados, como pode ser observado na Figura 4, estes códigos ainda podem ser utilizados.

Figura 4: Exemplos de código sujo e danificado.


Fonte: Adaptado pelo autor, baseado em Jesus (2018).

### 2.4.3 Utilização do *QR Code*

Conforme Xavier (2011), qualquer pessoa pode realizar a geração de um *QR Code*, isso se deve ao fato da sua utilização ser livre. Existem inúmeros *sites* que fazem a geração destes códigos, um dos usos mais comuns é com *links* para redirecionar um usuário que faz a leitura deste código a um site, como *blogs* e *sites* comerciais.

Coelho (2013), cita outros casos de uso do *QR Code*, como a televisão, onde o telespectador é conduzido para receber conteúdos extras das matérias mostradas. Pizzarias também iniciaram o uso para que o cliente ao ler o código possa realizar a compra de seus produtos. Outro caso famoso de uso é do *WhatsApp*, no qual utiliza de um *QR Code* para sincronizar o usuário na sua versão *web*.

E até mesmo para realizar pagamentos, conforme Rezende (2019), algumas empresas já disponibilizam o *QR Code* como método de pagamento, que tem por grande vantagem a segurança, pelo fato de não precisar estar com dinheiro em mãos para realizar a compra.

O grande sucesso do *QR Code* se deve por não possuir nenhum custo em sua geração. Além disso, os produtos que fazem sua leitura são de baixo custo. Um

destes dispositivos é o *Raspberry Pi*, um microcontrolador no qual pode ser conectado a outros inúmeros equipamentos, como câmeras e leitores que realizam a captura destes códigos e os decodificam.

## 2.5 Leitores de *QR Code*

Freitas (2017), define que para a leitura dos *QR Code* são necessários *Scanners* e/ou aplicativos, estes *Scanners* são divididos em três tipos como podemos ver a seguir:

- *Handy Scanner*: Este tipo de leitor faz a leitura dos códigos e em seguida são processados por um programa em um computador. Por ter pouco alcance em sua leitura, é recomendado o uso em ambientes menores.
- *Handy Terminal*: Neste *Scanner*, além de realizar a leitura dos códigos ele também realiza o armazenamento deles. Pode ser instalado em qualquer ambiente, e permite inúmeras formas de utilização.
- *Scanner* Fixo: Estes operam em conjunto com um computador conectados, e assim realizam a leitura dos códigos. É bastante utilizado para leitura de bilhetes em entradas de eventos, por exemplo.

Além destes *scanners*, existem inúmeros aplicativos para *smartphones* que realizam a leitura de *QR Codes* padronizados, necessitando apenas de uma câmera. Após a leitura, os dados são exibidos na tela do celular, caso for algum endereço *web*, pode ser redirecionado a este endereço (FREITAS, 2017).

Freitas (2017), cita alguns aplicativos que fazem essas leituras tanto para *Android*[2] e iPhone Operating System (*iOS*[3]), para *Android* é citado *Barcode Scanner*, *Norton Snap, QR Code Reader, QR Droid*, entre outros. Para *iOS* é citado *QR Reader for iPhone, QR Code by Scan*, Digitalização de código *QR*, entre outros.

[2] Android é o Sistema Operacional utilizado na maioria dos dispositivos móveis atuais.
[3] iOS é o Sistema Operacional utilizado nos dispositivos móveis da marca Apple.

# 3 TRABALHOS RELACIONADOS

## 3.1 Controle de acesso para estádio de futebol utilizando tecnologia NFC

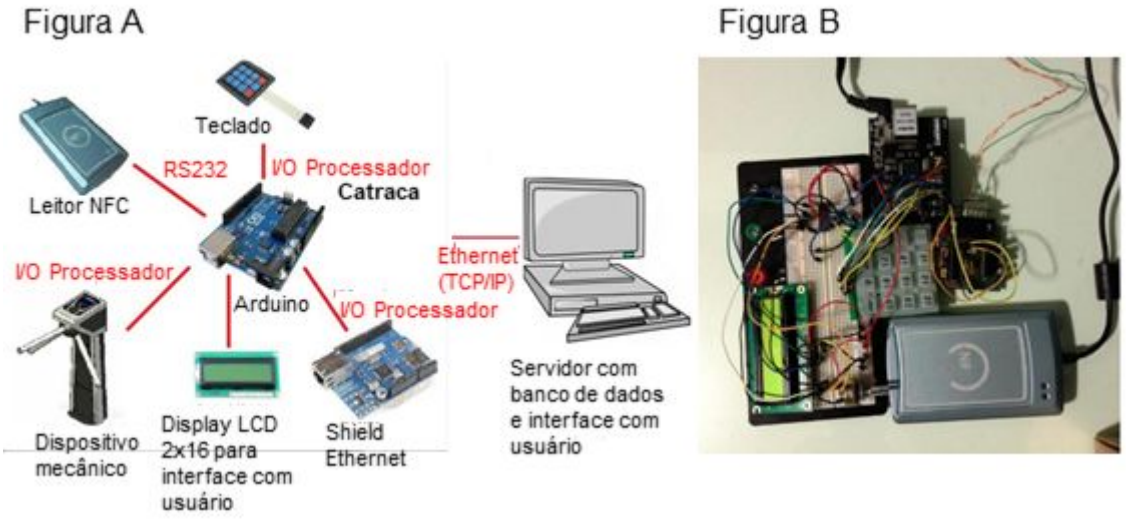
É um projeto de pesquisa elaborado por Palmeira e Fernandes (2013), este projeto tem como objetivo desenvolver um protótipo e um sistema que facilite o acesso aos estádios de futebol com a liberação das catracas com uma "*tag*" de *Near Field Communication* (NFC).

De acordo com Palmeira e Fernandes (2013), o protótipo foi desenvolvido utilizando uma placa microcontroladora *Arduino*, onde este é responsável por toda comunicação entre os dispositivos conectados a ele e o sistema proposto. Este sistema foi desenvolvido na linguagem C#, utilizando o banco de dados *SQL Server*, sendo este sistema responsável pela realização dos cadastros dos acessos e de fazer a verificação das validações.

O funcionamento é simples. O usuário passa sua tag no leitor NFC, que realiza o envio ao *Arduino* e este envia ao sistema, no qual verifica se está cadastrado, caso não esteja, é aberto a tela para realizar o cadastro desse usuário. Caso tenha cadastro, então é verificada a permissão de acesso, assim enviando ao microcontrolador a resposta e se possuir a permissão é liberado a catraca.

Ainda segundo Palmeira e Fernandes (2013), é especificado os dispositivos utilizados em um fluxograma, juntamente com o protótipo desenvolvido, como pode ser observado na Figura 5.

Figura 5: Figura A: Dispositivos utilizados, Figura B: protótipo desenvolvido.




Fonte: Adaptado pelo autor, com base em Palmeira e Fernandes (2013)

Após a realização de inúmeros testes com uma base de dados relativamente grande, tentando o acesso indevido e o acesso com permissão, Palmeira e Fernandes (2013), chegaram ao resultado que o protótipo tem uma confiabilidade de cerca de 98%, onde a taxa de erro foi de apenas 0,2%.

Outro ponto abordado por Palmeira e Fernandes (2013), foi o custo da ferramenta comparado a produtos similares, ficando na mesma faixa de preços e com o diferencial de ter um sistema para o controle dos acessos, no qual os outros produtos não possuem. Além disso, demonstrou extrema confiabilidade perante aos testes realizados.

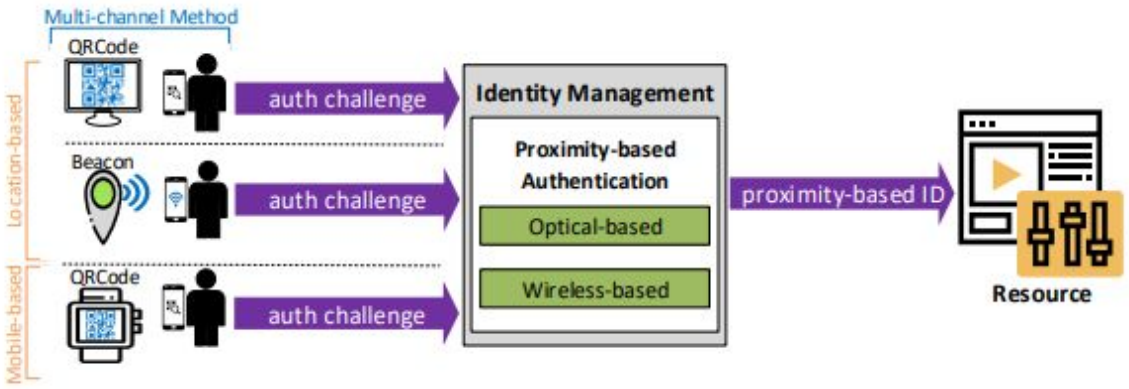
### 3.2 Modelo de autenticação multicanal baseada em proximidade

O trabalho realizado por Junior (2018), tem como objetivo a elaboração de uma ferramenta para autenticar usuários, que utiliza de técnicas multifatorial e multicanal, no qual ele se propôs a desenvolver um modelo utilizando três formas de autenticação por proximidade, sendo capaz de serem utilizados de forma individual ou em conjunto.

De acordo com Junior (2018), utilizou-se dos métodos baseados em rede *Wireless,* leitura óptica *e Wearable,* no qual estes trabalham de forma multicanal,

sendo assim, caso algum dos métodos estivesse comprometido, não afetaria o funcionamento do sistema como um todo. Para fazer essa autenticação foi utilizado um sistema de gestão de identidades (IdM), como apresentado na Figura 6 a seguir.

Figura 6: Demonstrativo do modelo proposto.




Fonte: Junior (2019).

Na definição de Junior (2018), o IdM é um serviço que realiza o controle das credenciais dos usuários e autenticação quando é chamado. Outra responsabilidade do IdM é constar na sua base de dados, o local físico de cada âncora e quem possui permissão para se autenticar nela.

Junior (2018), explica que para interagir com os usuários, foi desenvolvido um aplicativo *mobile*, sendo este responsável por comunicar-se com os demais componentes. Neste aplicativo os usuários possuem a liberdade de escolher qual método realizará a autenticação. Após escolher o método que quer se autenticar, o aplicativo vai procurar as âncoras próximas ao dispositivo. Ao encontrar uma âncora, a requisição é enviada ao IdM para realizar a autenticação.

Segundo Junior (2018), foram realizados testes com diferentes quantidades de dispositivos âncoras. Após a realização dos testes foi analisado que, com o componente *Beacon*, quanto mais dispositivos existirem mais lenta ficará a autenticação.

Nos testes com o *QR Code* esta proposta mostrou pouca diferença quando usados mais dispositivos. Isso se deve ao fato de que na requisição da autenticação já enviar o ID do local de autenticação, ao IdM (JUNIOR, 2018).

De acordo com Junior (2018), concluiu-se que para lugares com maiores quantidades de usuários o melhor método seria o *QR Code*. Mas Junior (2018), ressalta que para verificar a viabilidade real dos protótipos precisaria realizar testes num ambiente real, pelo fato da possível interferência gerada pelas redes *Wi-Fi.*

### 3.3 Autenticação de Dispositivos Móveis usando NFC

Este trabalho foi realizado por Ota (2016), da Universidade Estadual Paulista (UNESP), com o objetivo de propor um protótipo para realização de uma comunicação criptografada com uma aplicação utilizando cartões de *NFC*. Neste trabalho foram utilizados dois métodos de autenticação com cartão *NFC* sendo do tipo de armazenamento e do tipo *MIFARE DESFire*.

De acordo com Ota (2016), o primeiro método foi utilizado um cartão *NFC* de armazenamento, no qual este cartão contém armazenado uma chave compartilhada para a autenticação do aplicativo. O protocolo de autenticação inicia quando é aproximado o cartão ao dispositivo móvel, o aplicativo então lê a chave e manda ao servidor, indicando qual usuário e cartão serão utilizados.

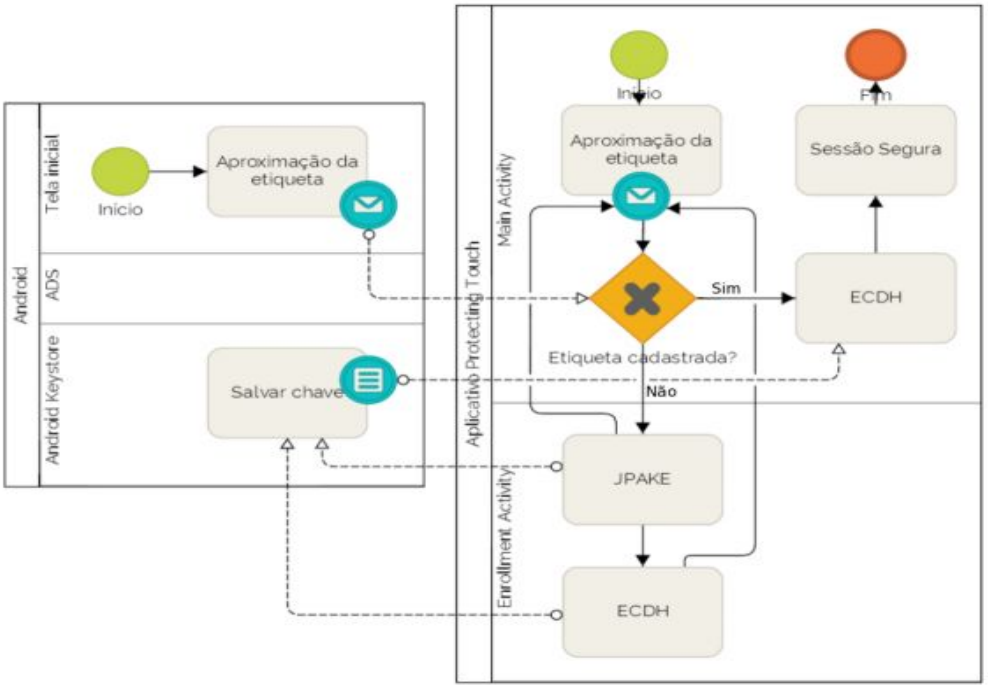
E então é iniciada a autenticação e, caso não encontre nenhuma chave autenticada, o aplicativo é encerrado. Caso for autenticado, continuam a comunicação de forma segura e criptografada (OTA, 2016).

Segundo Ota (2016), o método que utiliza o cartão *NFC MIFARE DESFire* implementa o protocolo de autenticação em três vias para autenticar com o servidor. Este tipo de cartão tem por diferencial autenticar a resposta da leitura, usada para ler arquivos é criptografada e autenticada, o aplicativo então usa este fato para obter o conteúdo dos dados de um arquivo criptografado pela chave da seção da comunicação entre aplicativo e servidor.

De acordo com Ota (2016), a etiqueta *DESFire* possui dois arquivos, o primeiro contém o identificador do usuário que é legível e o segundo a chave secreta

de longo prazo, compartilhada entre o servidor e o cartão, que só é legível no modo criptografado. Além disso, possui uma chave de autenticação mútua entre o cartão e o servidor sendo a autenticação e criptografia desta chave obrigatória para ler o segundo arquivo. A Figura 7 mostra o fluxograma do funcionamento destes protocolos.

Figura 7: Fluxograma do modelo proposto.




Fonte :OTA, 2016.

Ota (2016), explica que para medir a eficiência destes protocolos foram realizados alguns testes utilizando três diferentes celulares, todos compatíveis com alguns requisitos estabelecidos. Para Ota (2016), foi analisado que no quesito de usabilidade a proposta é extremamente considerável, mas que se deve considerar a internet usada, pois interfere diretamente na execução do aplicativo.

De acordo com Ota (2016), no quesito do uso dos recursos do aparelho celular, notou-se que esse uso é baixo, que varia conforme os protocolos, mas que mesmo em aparelhos mais antigos, esse consumo de recursos ainda é baixo. Portanto conclui-se que estes protocolos são capazes de substituir o uso de senhas

convencionais, que além de oferecer segurança, e velocidade na autenticação (OTA, 2016).

### 3.4 SAAS: uma Solução de Autenticação para Aplicativos de Smartphones

É uma proposta elaborada por Fernandez *et al*. (2019), da Universidade Federal do Rio Grande do Norte (UFRN) e que tem como principal objetivo o desenvolvimento de um modelo de autenticação seguro de um único fator, e que não interfira na usabilidade do acesso aos locais.

Fernandez *et al*. (2019), baseiam-se no sistema de controle de acessos do Serviço Social do Comércio do Rio Grande Sul (SESC-RS), no qual utilizam um mecanismo de autenticação de acesso de único fator. Além disso, os *QR Codes* gerados para o acesso são estáticos, com o Cadastro de Pessoa Física (CPF) de cada pessoa.

De acordo com Fernandez *et al*. (2019), este modelo de autenticação é facilmente burlado. Com isso eles propõem uma solução ainda em único fator, para não interferir na usabilidade dos usuários, porém exige que os QR Codes gerados sejam diferentes para cada acesso daquele usuário.

Fernandez *et al*. (2019), comenta que a solução é composta por dois protocolos. O primeiro que faz a identificação do usuário com seu dispositivo móvel, com o propósito de somente um dispositivo conter a identificação do usuário. O segundo é responsável pela geração dos QR Codes descartáveis, chamados de *One-Time Authentication Codes*, onde cada QR Code seria válido para apenas uma autenticação.

Fernandez *et al*. (2019), para realizar o controle dos dispositivos utilizados pelos usuários é utilizado a Identificação Internacional de Equipamento Móvel (IMEI), para a geração das chaves que compõem o QR Code, e assim a cada acesso é gerado um novo QR Code.

Para Fernandez *et al*. (2019), a proposta desenvolvida entrega QR Codes difíceis de serem clonados, e além disso são descartáveis, podendo serem utilizados

somente uma vez. Outro ponto vantajoso que o autor aponta é a rapidez com que o processo de autenticação ocorre.

### 3.5 Transposição da Autenticação Federada para uma Solução de Controle de Acesso Físico no contexto da Internet das Coisas

Este trabalho elaborado por Silva *et al*. (2018), tem como objetivo elaborar uma solução de autenticação de acesso físico utilizando o padrão *Security Assertion Markup Language* (SAML). Este padrão é utilizado para atributos baseados em *Access-based Access Contro*l (ABAC), e no padrão *Universal Authentication Framework* (UAF) da *FIDO Alliance*, responsável por criptografar a chave pública, e com isso a autenticação ocorre sem necessitar de uma senha.

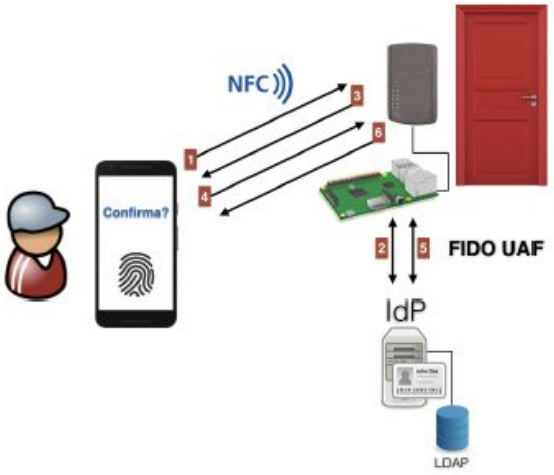
Silva *et al*. (2018), explicam que a solução está dividida em quatro elementos, sendo o primeiro um aplicativo *Android*, que realiza a autenticação dos usuários utilizando o protocolo FIDO, se comunicando via NFC com o controlador de acesso físico, e exibe o retorno desta autenticação para o usuário.

O segundo elemento é o controlador físico, no qual foi utilizado um *Raspberry Pi,* que realiza essa comunicação com o *smartphone* e com uma trava magnética. O terceiro elemento utilizado é o controlador lógico, que está embarcado no Raspberry Pi, que é responsável por executar as regras de negócio.

O quarto elemento é denominado Multi-Factor Provider (MFaP), o qual utiliza do protocolo FIDO UAF para realizar a autenticação dos usuários, apesar de não explorado nesta solução, este elemento poderia realizar a autenticação com múltiplos fatores de autenticação.

Para Silva *et al*. (2018), a solução proposta demonstrou viabilidade e aplicabilidade. A Figura 8 demonstra como seria o processo de autenticação de um usuário em um local físico.

Figura 8: Demonstração do funcionamento da solução proposta.




Fonte: Silva *et al*. (2018).

Como pode ser observado, a autenticação ocorre a partir do NFC do smartphone, onde caso o acesso seja liberado, irá mandar um sinal para a Raspberry Pi, que abrirá a porta.

# 4 MATERIAIS E MÉTODOS

Neste capítulo são abordados os processos e as ferramentas para o desenvolvimento do trabalho, bem como os métodos utilizados para a elaboração da solução deste projeto.

Quanto aos procedimentos, a presente pesquisa é classificada como pesquisa experimental, pois está condicionada ao desenvolvimento de um protótipo. Segundo Barros e Lehfeldl (2007), a pesquisa experimental se trata de observar, interpretar a atuação das variáveis sobre os objetos de estudo.

Barros e Lehfeldl (2007), ainda complementam que definir como procedimento o método experimental, significa determinar as hipóteses para a realização do experimento. Verificando o comportamento da solução *AuthToken* na validação dos acessos.

Para Cervo et al. (2006), a pesquisa experimental tenta deduzir de que modo e porque ocorre um acontecimento. Para analisar esses acontecimentos faz-se o uso de ferramentas e métodos que possam compreender a relação entre as variáveis do objeto de estudo. São monitoradas, portanto os usuários que interagem com a solução, estes realizaram interações de tentativas de acesso possuindo ou não permissão, com isso será analisado o comportamento da solução na autorização destes acessos.

Além da pesquisa experimental, esse trabalho envolve a pesquisa bibliográfica para o seu desenvolvimento, pelo fato de ser preciso averiguar outras

bibliografias similares, para entender os estudos já feitos e quais seus resultados. Segundo Cervo et al. (2006), procura-se em outros trabalhos realizados contribuições sobre o assunto desejado, para obter conhecimentos prévios e domínio do assunto.

No que se refere aos objetivos, essa pesquisa é classificada como exploratória, pois se trata de estudar uma tecnologia atual, que é pouca utilizada nesta area, Barros e Lehfeldl (2007), explicam que a pesquisa exploratória se delimita a definir um objeto de pesquisa e explorar-lo buscando conseguir novas informações sobre o estudo. Este tipo de pesquisa, deve-se estar organizado, pois as ponderações sobre o objeto de estudo abrangem inúmeros aspectos.

Em relação ao tipo de pesquisa, este trabalho é definido como relatório técnico, em razão de ser mais conveniente em trabalhos em que é desenvolvido um produto. De acordo com a Norma Brasileira NBR 10719 (2011), é um documento no qual expõe os resultados e os avanços obtidos na análise e desenvolvimento da pesquisa. O relatório ainda apresenta informações de forma organizada e realiza conclusões sobre a pesquisa realizada.

### 4.1 Tecnologias utilizadas

O objetivo do presente trabalho é o desenvolvimento de um dispositivo que visa otimizar o processo de autenticação e autorização com base em *QR Code* e que possa, além de gerenciar o processo de autenticação e autorização, fornecer aos administradores uma consulta dos registros de acessos.

Para que tal dispositivo possa ser implementado, serão utilizadas as tecnologias *JavaScript*, que é utilizada juntamente com a *Api Google Charts* para a geração do *QR Code* em um aplicativo multiplataforma*, com Python* para realizar a leitura e decodificação dos códigos e também com o *Raspberry Pi* acoplado de uma câmera para realizar a captura das imagens dos códigos gerados.

Como *framework* para o desenvolvimento do aplicativo para os usuários foi utilizado o *Ionic*. Este aplicativo utiliza  do tipo REST para realizar a comunicação com o banco de dados, desenvolvidas na linguagem de programação PHP. A

escolha pela utilização deste *framework* se deve ao fato deste permitir construir aplicativos multiplataforma.

### 4.1.1 *Python*

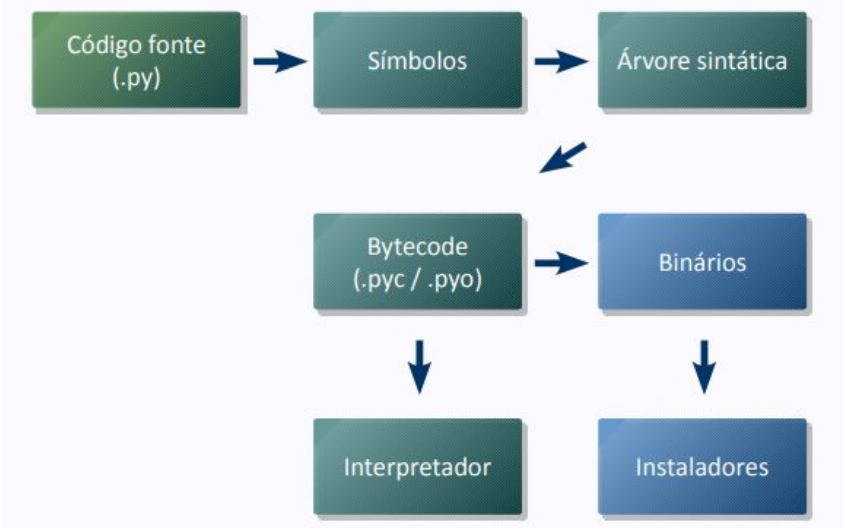
De acordo com Borges (2010), a linguagem de programação *Python* foi criada em 1990 por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda. É uma linguagem de programação de alto nível e fácil entendimento, orientada a objetos e que possui inúmeras e *frameworks* que podem ser interligados com *Python*.

Para Borges (2010), a linguagem de programação Python pode ser utilizada como linguagem principal de um sistema ou como *scripts* em outros sistemas, automatizando funções e criando novas ferramentas. Além disso, é possível utilizar *Python* em conjunto com outras linguagens.

> A linguagem é interpretada através de *bytecode* pela máquina virtual *Python*, tornando o código portável. Com isso é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar direto do código fonte.(BORGES, 2010, p.13).

A Figura 9 ilustra como é feita a compilação destes códigos.

Figura 9: Processo de compilação da linguagem *Python*.




Fonte: Borges (2010).

Para o desenvolvimento do protótipo também foi considerado o critério da facilidade de encontrar bibliotecas que auxiliam no tratamento de imagens e decodificação dos *QR Codes* com base na linguagem *Python*. Além disso, a sua escolha se deve ao fato de ser uma linguagem de sintaxe de fácil entendimento e ágil na produção de resultados.

### 4.1.2 Hardware

O hardware utilizado para realizar a leitura dos *QR Codes* será um *Raspberry* Pi, conectado a uma câmera, que realizará a leitura do código. A escolha pelo *Raspberry Pi*, se deve ao fato deste dispositivo possuir *Wi-Fi* acoplado, sem a necessidade de uma *Shield,* além de ser um *hardware* robusto que permite inúmeras aplicações.

#### 4.1.2.1 *Raspberry Pi*

O *Raspberry Pi* é um microcontrolador de baixo custo que possui inúmeras funcionalidades, no qual podem ser conectados monitores, teclados, mouses, entre outros dispositivos (LUZEMBO, 2017). Este microcontrolador pode conter um sistema operacional *Linux*, inclusive com suporte à interface gráfica. A Quadro 2 a seguir, faz um comparativo entre as versões do *Raspberry Pi*.

Quadro 2: Comparativo entre as versões do *Raspberry Pi*.

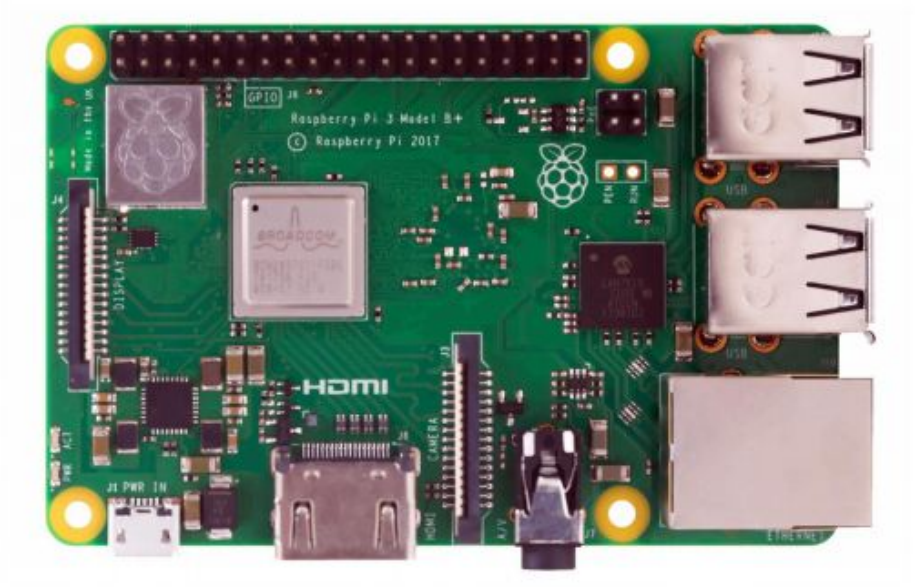
| ***Raspberry Pi*** | ***SoC*** | ***CPU*** | ***RAM*** | **Porta *USB*** | ***Ethernet*** | ***Wireless/Bluetooth*** |
|---|---|---|---|---|---|---|
| *Model* A+ | BCM2835 | 700Mhz | 512MB | 1 | ✗ | ✗ |
| *Model* B+ | BCM2835 | 700Mhz | 512MB | 4 | ✓ | ✗ |
| 2 *Model* B | BCM2836 | 900Mhz | 1GB | 4 | ✓ | ✗ |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 *Model* B | BCM2837 | 1200Mhz | 1GB | 4 | ✓ | ✓ |
| Zero | BCM2835 | 1000Mhz | 512MB | 1 | ✗ | ✗ |
| ZeroW | BCM2838 | 1000Mhz | 512MB | 1 | ✗ | ✓ |

Fonte: Adaptado pelo autor, com base em Luzembo (2017).

O modelo de microcontrolador utilizado no projeto é o Raspberry Pi 3+, que possui um processador de 4 núcleos com 1,4 GHz de 64 bits e 1 *GigaByte* (GB) de *Random Access Memory* (*RAM*). Já para as questões de conectividade, possui quatro portas *Universal Serial Bus* (*USB*), uma interface *Fast Ethernet, Bluetooth, Wireless Local Area Network (WLAN)*, saída *High-Definition Multimedia Interface* (*HDMI)*, além de 40 pinos de entrada e saída, permitindo conectar diversos outros dispositivos. A Figura 10 a seguir ilustra o dispositivo utilizado.

Figura 10: Ilustrativo do *Raspberry Pi* 3 modelo B+




Fonte: RASPBERRY PI FOUNDATION (2020).

Inicialmente, o *Raspberry Pi* foi proposto como uma solução de computador popular, executando uma versão do sistema operacional *Linux*, com interface gráfica e pacotes de aplicativos de código aberto. Seu sucesso, no entanto,

> abriu espaço para várias outras aplicações especialmente em sistemas embarcados. (OLIVEIRA, 2017, p. 37).

Segundo Oliveira (2017), o sistema operacional *Linux* oferece uma gama de programas de desenvolvimento, para inúmeras linguagens de programação, com ou sem interface gráfica. Também possui diversas Application Programming Interface ([4] ), que podem realizar a integração com diversos dispositivos, assim como teclado, câmeras, sensores, leitores biométricos, *NFC*, entre outros.

### 4.1.3 Aplicativo

Para José (2016), o a *Ionic* é um dos *frameworks* mais utilizados do mundo para o desenvolvimento de aplicativos *mobile*, além de ser *open source* é composto por ferramentas *web*, como *Hypertext Markup Language version 5* (HTML5)*, Cascading Style Sheets* (CSS) *e JavaScript,* para o desenvolvimento de aplicativos híbridos[5] e de multiplataformas.

A utilização deste *framework* visa construir um aplicativo mobile de multiplataforma, sendo assim, possibilitando ser utilizado por todos os públicos de *smartphones*, seja usuários *Android* ou *IOS*.

De acordo com José (2016), o *Ionic* tem como grande diferencial a busca por desempenho e agilidade no desenvolvimento de aplicativos. O *Ionic* também faz uso do *Apache Cordova* e um *command-line interface* (CLI) robusto, que permite a criação, realizar testes e publicar o aplicativo com poucos comandos.

Na definição de Dimes (2015), *JavaScript* é uma linguagem de *scripting*, utilizada na comunicação entre as páginas *web*, popularmente a mais utilizada para estes fins. Podem ser utilizadas de forma integrada a páginas HTML, para realizar o trabalho pesado.

De acordo com Dimes (2015), apesar de realizar estes trabalhos mais pesados com sintaxe de códigos simples, estes códigos podem se tornar complexos, pois existem estruturas de programação mais complexas que requerem um estudo

[4] são um conjunto de códigos, que conectam funcionalidades com a aplicação a ser desenvolvida.
[5] Aplicativos hídricos utilizam tecnologias *web* e nativas.

mais aprofundado, além de apresentar diferenças no suporte à linguagem entre os navegadores *WEB*.

Por fim, o aplicativo utiliza a *Api Google Chart* para a geração dos *QR Codes* que é disponibilizada gratuitamente pela Google. Além desta *Api*, foram desenvolvidas quatro  do tipo *REST* em *PHP* para a realização do envio de informações ao banco de dados.

Estas tecnologias foram escolhidas visando criar um aplicativo rápido, leve e proporcionar fácil usabilidade ao usuário. Além disso, facilitam no desenvolvimento da aplicação.

# 5 DESCRITIVO DO PROTÓTIPO

Neste capítulo, são abordados os processos e as ferramentas para o desenvolvimento do trabalho, bem como os métodos utilizados para a elaboração da solução deste projeto.

## 5.1 Visão geral

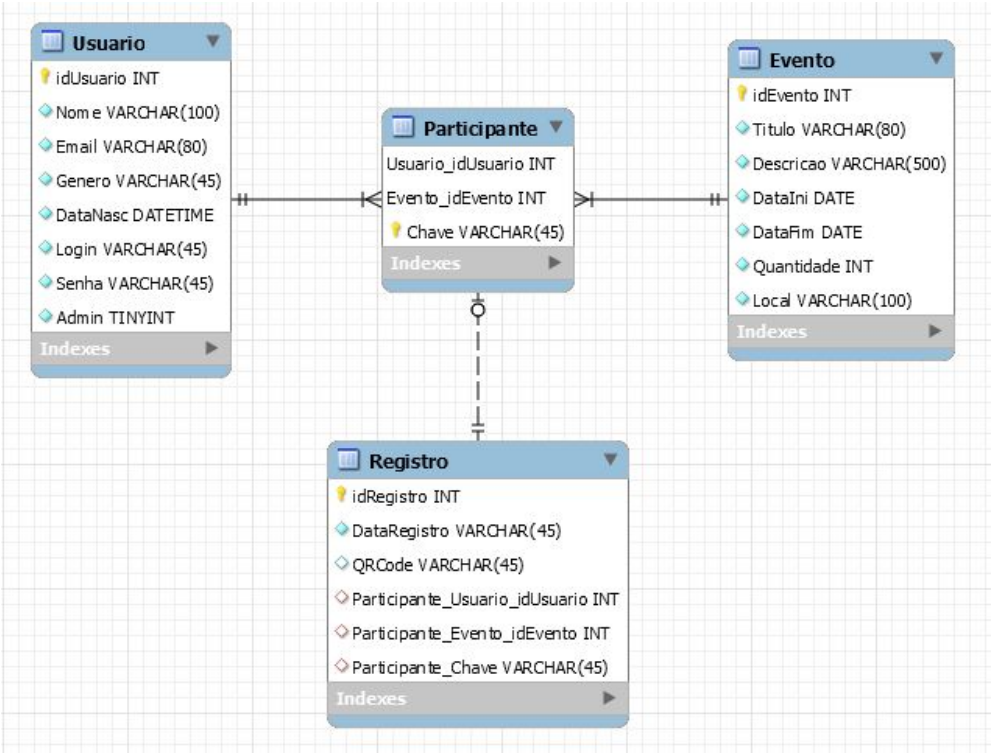
A solução AuthToken visa facilitar e fornecer gerenciamento no controle de acessos a eventos ou a lugares privados. O aplicativo é responsável pela realização dos cadastros de usuários, eventos e também por vincular estes usuários aos eventos, permitindo consultar quem acessou os eventos e em qual horário ocorreu.

A validação destes acessos ocorre em um Raspberry Pi, que com uma câmera, captura os códigos e os decodifica. Em seguida ele valida o usuário e confirma se este usuário tem permissão para acessar o evento, gerando um *log*. Deve, portanto, ser capaz de gerenciar todo o processo de validação dos acessos.

## 5.2 Diagrama ER

O presente protótipo foi elaborado para ser ágil, simples e viável economicamente. Orientado por essa premissa, a estrutura do banco de dados é estabelecida da seguinte maneira: uma tabela de usuários, onde estes podem participar de vários eventos, um tabela de eventos, que pode possuir uma quantidade X de usuários. Com esta ligação foi criada outra tabela, chamada de participantes e, para registrar o acessos destes, é criada a tabela registro. Como evidenciado na Figura 11 a seguir que apresenta o diagrama ER.

Figura 11: Diagrama ER da solução.



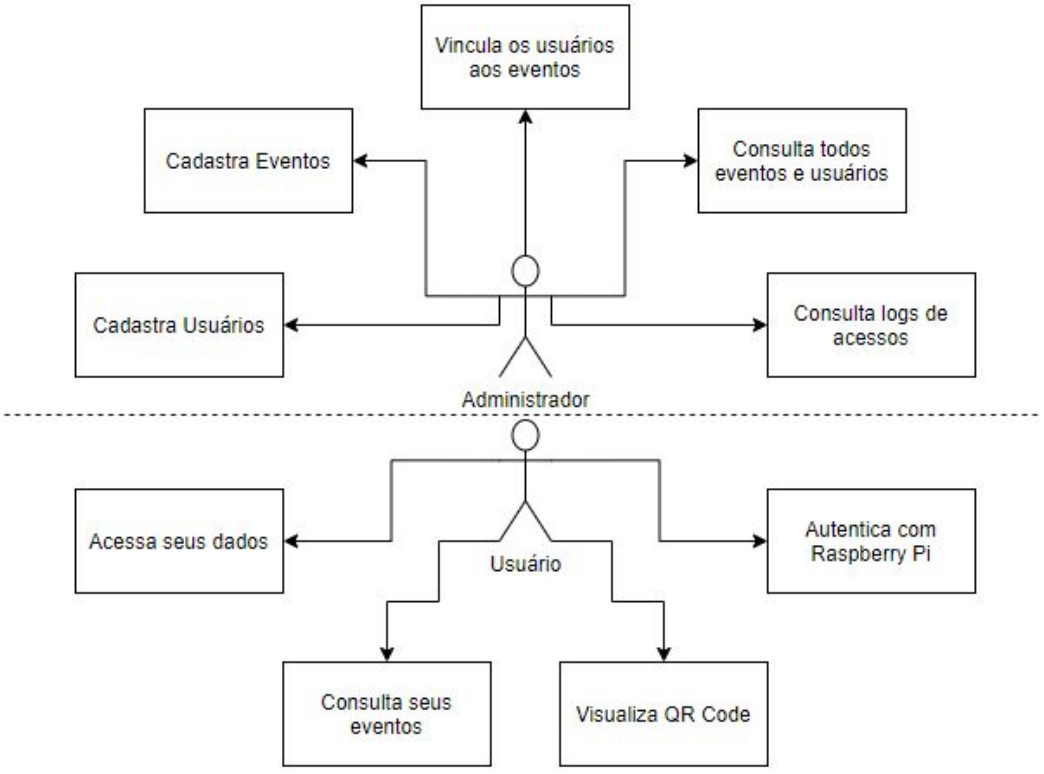

Fonte: Autor (2020).

Como pode ser observado, o banco de dados é relacional, ou seja, com a sua estrutura de campos definida e com as tabelas mantendo um relacionamento. O diagrama ER foi planejado de maneira simples, sem ramificar em demasia as tabelas.

### 5.3 Estrutura e diagrama dos casos de uso

Para demonstrar as interações dos dois perfis de usuários com o sistema, foram mapeados os processos que cada um pode realizar e também elaborado um diagrama dos casos de uso, demonstrado na Figura 12. Nele podemos observar os dois atores do sistema, o administrador que possui a permissão para cadastrar os demais usuários, cadastros de eventos e vincular estes usuários aos eventos, o que é necessário para o correto funcionamento do gerenciamento dos acessos.

Figura 12: Diagrama dos casos de uso.




Fonte: Autor (2020).

Destaca-se que, além dos acessos, também é possível consultar quem registrou o acesso pelo protótipo. Já o ator usuário possui acesso somente aos seus dados cadastrais, e os eventos no qual está vinculado ou já participou, além da tela de visualização do *QR Code*.

### 5.4 Desenvolvimento do *web service* e

Para a realização da comunicação entre o aplicativo e o dispositivo móvel, foi criado um conjunto de cinco  em PHP, sendo elas *login*, evento, participante, usuário e registros. Estas  ficam em um servidor *Apache*, para que possam ser acessadas pela rede. Para isso foi utilizado o *Xampp*, que é um programa que possui vários pacotes de servidor, incluindo o *Apache,*com suporte ao PHP.

Após instalar o *Xampp*, só é preciso iniciar os serviços do *Apache* e do *Mysql*. As  foram divididas para cada tabela do banco de dados, e cada uma delas possui seus métodos de requisições (GET, POST, PUT e DELETE).

Para que estas  possam fazer essas requisições é preciso permitir acesso a estes métodos e outras permissões necessárias no cabeçalho da Api, os chamados "*headers*", como pode ser observado na figura 13 a seguir, além da conexão com o banco de dados.

Figura 13: *Headers* das .

```php
<?php
  header('Access-Control-Allow-Origin: *');
  header('Access-Control-Max-Age: 86400');
  header("Access-Control-Allow-Credentials: true");
  header("Access-Control-Allow-Methods: POST, GET, DELETE, PUT, OPTIONS");
  header("Access-Control-Allow-Headers: Content-Type, Authorization, X-Requested-With");
  header("Content-Type: application/json; charset=utf-8");

    try{
        $conn = new mysqli('127.0.0.1','root','','auth');
    }catch( Exception $e){
        echo 'Error connect';
    }
```

Fonte: Autor (2020).

A partir disso, basta que os serviços do aplicativo ao comunicarem-se com estas  via HTTP, informem o método a ser requisitado.

### 5.5 Desenvolvimento da aplicação mobile

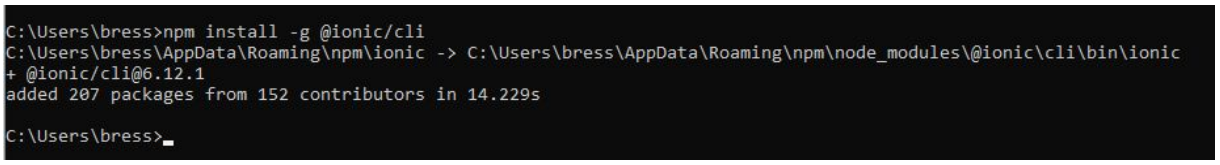
Nessa seção são demonstradas as interações com os usuários, bem como o desenvolvido uma aplicação *mobile*, no qual foi utilizado a IDE Visual Studio Code, juntamente com o *framework Ionic*.

#### 5.5.1 Framework Ionic

O primeiro passo foi fazer a instalação dos recursos necessários para o funcionamento deste *framework* que utiliza a linguagem de programação JavaScript, portanto é necessário instalar o Node.js para que possa rodar as aplicações no

próprio navegador. Após sua instalação, abrimos o *Prompt de Comando*, e é feita a instalação do Ionic cliente com o comando que é mostrado na Figura 14 a seguir.

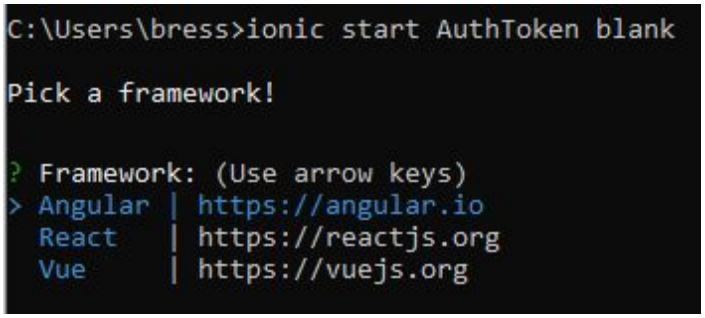
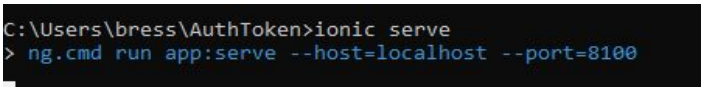
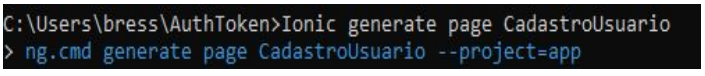
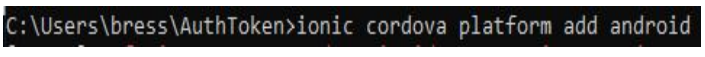
Figura 14: Instalação do Ionic




Fonte: Autor (2020).

Com o Ionic instalado, toda a etapa de criação do projeto envolvendo páginas e serviços para rodar a aplicação, incluindo a adição das plataformas Android, IOS e Windows, e a geração do aplicativo executável, é feito pelo próprio terminal. O Quadro 3 demonstra os principais comandos utilizados e uma breve explicação dos mesmos.

Quadro 3: Lista dos principais comandos utilizados no *Ionic*

| | |
|---|---|
|  | *Ionic start* é usado para a criação do projeto, onde existem 3 opções de layout, *blank, tabs e side menu.* Em sequência irá pedir em qual *Framework* irá trabalhar *Angular, React ou Vue*. |
|  | *Ionic serve* é usado para rodar a aplicação localmente. |
|  <br>  | *Ionic generate* é usado para a criação de páginas, serviços e demais componentes. |
|  | *Ionic cordova platform* é usado para adicionar as plataformas no qual o aplicativo vai ser construído. |

| | |
|---|---|
| `C:\Users\bress\AuthToken>ionic cordova build android` | *Ionic cordova build* é usado para a criação do aplicativo na plataforma definida. |

Fonte: Autor (2020).

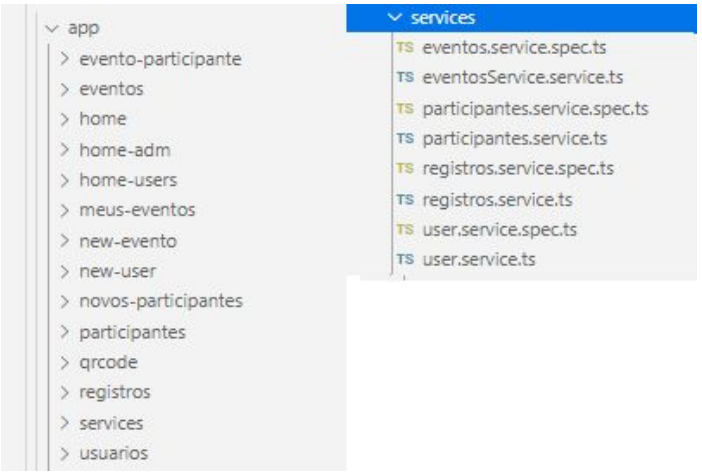
### 5.5.2 Estrutura do aplicativo.

Após criado o projeto do aplicativo, foi utilizado os comandos citados no tópico anterior para criar as páginas de cadastro de usuários, tela com os dados do usuário, listagem dos usuários, cadastros do eventos, listagem dos eventos, listagem dos eventos do usuário que logou, página de participantes, a página de visualização do *QR Code* e a página com os registros dos acessos realizado.

A tela inicial do aplicativo é onde os usuários irão realizar o login de acesso, sendo redirecionados para o menu inicial. Este redirecionamento é feito pelo tipo de usuário, onde o usuário com perfil de administrador acessa a página com mais opções de cadastros e consultas do que o usuário padrão.

Por fim, também foram criados quatro serviços, que são: UserService, EventosService, ParticipanteService e RegistrosService, utilizados para executar as que fazem as requisições com o banco de dados. A figura 15 a seguir demonstra a estrutura interna do projeto da aplicação.

Figura 15: Estrutura das páginas da aplicação e os serviços.




Fonte: Autor (2020).

Estes serviços são responsáveis por definir qual a requisição será feita na Api, além disso possuem diferentes funções para o mesmo método, como por exemplo o serviço UserService, que contém a função *login*, do método GET, que retorna somente um usuário, mas também possui a função getAll, também do método GET, que retorna todos os usuários cadastrados.

### 5.5.3 Desenvolvimento do aplicativo

Com a estrutura da aplicação definida, parte-se para o desenvolvimento do aplicativo, iniciando pela tela de *login*. Cada usuário, ao se cadastrar, terá um login e uma senha para se conectar e, a partir disso, terá acesso às funções do aplicativo, seja com perfil de administrador ou de usuário padrão. A Figura 16 demonstra a tela inicial de acesso ao aplicativo.

Figura 16: Tela de login.


Fonte: Autor (2020).

A partir de um login válido identifica o perfil associado à credencial, aplicando as permissões apropriadas. Isso ocorre a partir de uma requisição para a Api do usuário, como pode ser observado na Figura 17.

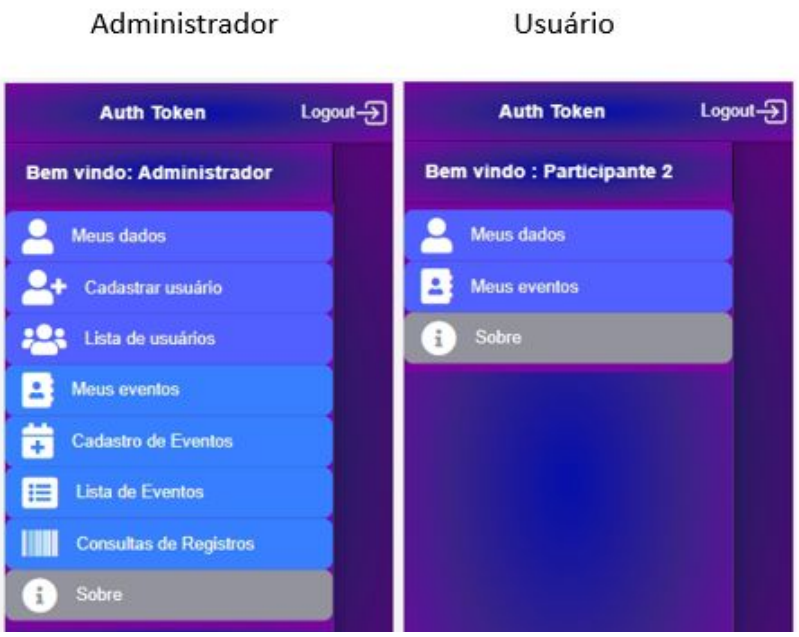
Figura 17: Função login

```
this.service.login(this.login, this.senha).subscribe(response => {
  this.loadingCT.dismiss();
  this.users = response
  if(this.users != null){
    if(this.users.admin == 1){
      this.navCT.navigateRoot('/home-adm/'+ this.users.id);
    }else{
      this.navCT.navigateRoot('/home-users/'+ this.users.id);
    }
  }
  })
```

Fonte: Autor (2020).

A partir desta validação, a tela inicial correspondente ao tipo do usuário é iniciada, conforme demonstrado na Figura 18.

Figura 18: Telas iniciais dos respectivos tipos de usuários.



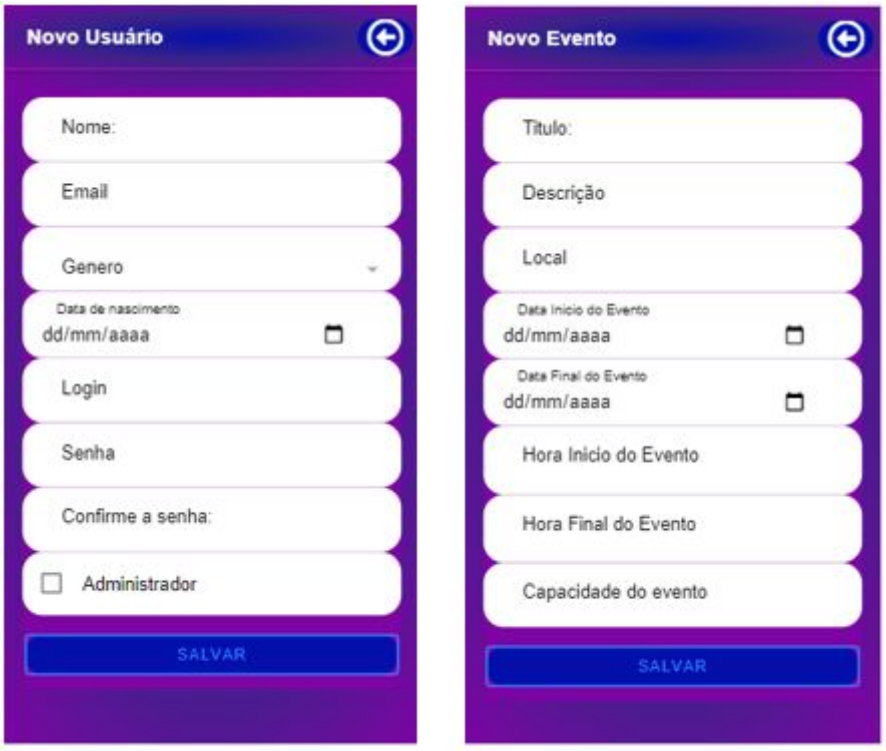

Fonte: Autor (2020).

A partir deste ponto, pode-se seguir para as telas de cadastros de usuários, eventos e relatórios. Estas telas solicitam somente algumas informações básicas para a realização do seu cadastro, onde esses dados são enviados para um serviço

em formato JSON, este serviço é quem enviará este JSON para o método HTTP requisitado.

O cadastro dos usuários destina-se ao preenchimento de dados pessoais tais como: nome, e-*mail*, gênero, data de nascimento, *login*, senha e informar se o usuário é administrador, enquanto que, para cadastro dos eventos são solicitados título, descrição, local, data e hora inicial e final, e a capacidade de usuários que podem participar do evento. A figura 19 a seguir ilustra as telas de cadastros citadas.

Figura 19: Telas de cadastros de Usuário e Eventos.



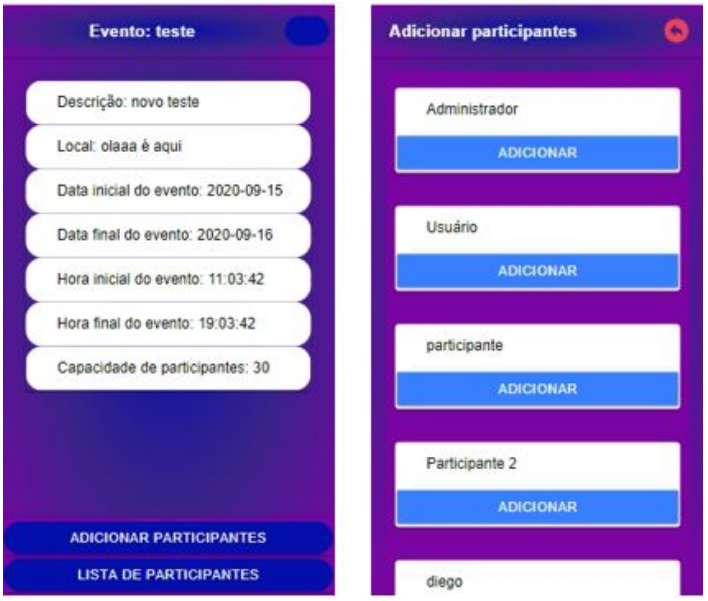

Fonte: Autor (2020).

Com os cadastros finalizados, foram desenvolvidas duas outras telas, destinadas para a obtenção das listagens desses registros, podendo adicionar, onde redireciona para a tela de cadastros novamente, editar os registros cadastrados e deletar. Essa listagem apresenta somente alguns dados do cadastro, para não sobrecarregar a visualização em tela.

Os eventos têm um diferencial, como pode ser observado na Figura 20, que permite abrir a tela com todos os dados de um evento. Também possui outra opção

para que se possa vincular usuários ao evento em questão. No botão “Adicionar participantes”, irá mostrar uma lista com os usuários que ainda não participam do evento, sendo possível adicionar novos usuários ao evento.

Figura 20: Tela do evento selecionado, e listagem dos usuários.




Fonte: Autor (2020).

Por outro lado, os usuários que não são administradores, possuem acesso somente aos eventos que estão vinculados. Para isso, dentro do serviço dos eventos, possui uma função específica para esta solicitação, como apresenta a Figura 21, a chamada desta função do serviço e a função deste serviço respectivamente.

Figura 21: Chamada do serviço e função getMeusEventos.

```
this.service.getMeusEventos(this.id).subscribe(response => {
this.eventoslist = response;
});

getMeusEventos(idUser: String){
  return this.http.get<[Eventos]>(this.url+ '?idUser=' + idUser);
}
```

Fonte: Autor (2020).

A função getMeusEventos solicita um método GET para a *Api*, passando o id do usuário logado, que retornará somente os eventos relacionados ao mesmo. Para estes usuários, ao selecionar o evento irá abrir a tela de visualização do *QR Code* para acessar o evento.

Este código é gerado a partir de uma função que gera caracteres aleatórios, em seguida é executada uma chamada ao serviço do participante que, por sua vez, retorna os dados do participante, com uma chave. A Figura 22 demonstra como é utilizada esta função.

Figura 22: Função da geração do *QR Code*

```
  geraQR(idEvento: String){
    this.chave= this.generateID(10);
    this.servicePart.getQR(idEvento,this.id).subscribe(response => {
      this.chaveParcial =response.chave;
      this.participante = response;
      this.chave = this.chave+"_"+this.chaveParcial;
      console.log(this.chave);
      this.modalCrtl.create({
        component: QrcodePage,
        componentProps: {
          'chave': this.chave
      }
      }).then( modal => modal.present());
      });
}
//funções paara gerar chave aleartoria.
dec2hex(dec){
  return ('0'+ dec.toString(16)).substr(-2);
}
generateID(len){
  var arr = new Uint8Array((len || 40)/ 2);
  window.crypto.getRandomValues(arr);
  return Array.from(arr,this.dec2hex).join('');
}
```

Fonte: Autor (2020).

Esta chave do participante é gerada pela mesma função generateID no momento em que o usuário é vinculado ao evento. A chave que gera o QR Code, portanto, é composta por duas partes. Esta segunda chave que é gerada ao visualizar o código, é uma chave dinâmica, ou seja, cada vez que é selecionado este evento irá gerar um novo QR Code, diferente do mesmo evento que foi visto minutos antes, por exemplo.

Após ter a concatenação destas duas chaves, é feita a chamada da tela que visualiza o QR Code, passando por parâmetro esta chave. Em seguida, nesta tela, é armazenado em uma variável a Uniform Resource Locator (URL) da *Api Google chart* que juntamente com a chave que gera o *QR Code*, como é apresentado na Figura 23.

Figura 23: Recebimento da chave como parâmetro e URL da *Api Google Chart*

```
export class QrcodePage implements OnInit {
  chave: String;
  public urlQR: String = "https://chart.googleapis.com/chart?chs=200x200&cht=qr&chl=";
  constructor(private navpar: NavParams,
    ) {
      this.chave = navpar.get('chave');
    }

  ngOnInit() {
    this.urlQR = this.urlQR +""+ this.chave;
    console.log(this.urlQR);
  }
```

Fonte: Autor (2020).

Por fim esta variável contendo a URL é passada para uma imagem, que representa na tela o *QR Code*. A aplicação deste método de gerar uma chave nova toda vez que é visualizado foi utilizado para evitar que estes códigos sejam clonados.

A Figura 24 ilustra a exibição de um *QR Code* gerado a partir da junção destas duas chaves. Como pode ser observado, existe uma borda na cor branca ao redor do código, esta foi utilizada para aumentar o destaque deste código, com isso em ambientes com pouca iluminação, o próprio brilho do dispositivo já irá auxiliar na leitura deste *QR Code*.

Figura 24: *QR Code*.


Fonte: Autor (2020).

Esse *QR Code* gerado deve ser utilizado pelo usuário no processo de autenticação do evento. O dispositivo então captura esse código a partir da câmera que está acoplada a um *Raspberry Pi* para validar o seu acesso ao evento.

### 5.6 Desenvolvimento do protótipo de leitura dos códigos

Para o desenvolvimento do protótipo de leitura dos *QR Codes*, foi utilizado um *Raspberry Pi e uma* câmera acoplada, conforme ilustrado na Figura 25. O processo de decodificação destes códigos é realizado a partir da biblioteca *OpenCV*.

Figura 25: Protótipo para leitura do QR Code.

Fonte: Autor (2020)

Para a elaboração da decodificação destes *QR Codes*, foi utilizado a linguagem de programação *Python*, visto a sua larga aplicação na área de processamento de imagens. Para a utilização da biblioteca *OpenCV*, foi necessário instalar alguns pacotes ou serviços, conforme demonstrado na Figura 26.

Figura 26: Instalação dos pacotes para utilização do *OpenCV*.

```
sudo apt-get install -y build-essential cmake pkg-config
sudo apt-get install -y libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install -y libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install -y libxvidcore-dev libx264-dev
sudo apt-get install -y libgtk2.0-dev
sudo apt-get install -y libatlas-base-dev gfortran
sudo apt-get install -y python3-dev python3-dev
```

Fonte: Autor (2020).

O desenvolvimento do leitor dos *QR Codes* foi dividido em dois arquivos de códigos. O primeiro *QRCode.py*, é responsável pela leitura e decodificação dos *QR Codes* e o segundo *Conexao.py*, responsável por validar o acesso junto ao banco de dados. Esse segundo arquivo é chamado pela função gravarRegistro, presente no *QRCode.py*.

Figura 27: *QRCode.py*, arquivo para a decodificação dos QR Codes.

```
import cv2
from time import sleep
from conexao import *

cap = cv2.VideoCapture(0)
detector = cv2.QRCodeDetector()
qrcode = None
d2 = None
while True:
    _, img = cap.read()
    data, bbox, _ = detector.detectAndDecode(img)
    if(bbox is not None):
        if data :
            d1,d2 = data.split("_")
            print("data: ", d2)
            if d2 != qrcode:
                gravarRegistro(d2)
            qrcode = d2
            d = None
            data is False
            bbox is None

    cv2.imshow("code detector", img)
    if(cv2.waitKey(1) == ord("q")):
        break
cap.release()
cv2.destroyAllWindows()
```

Fonte: Autor (2020).

Neste primeiro arquivo é onde é utilizada a biblioteca *OpenCV*, como apresentado na Figura 27, onde inicialmente é importado as bibliotecas necessárias para o funcionamento, sendo elas a CV2 do *OpenCV*, *time* capturar a data e hora que será feito o registro e o arquivo *Conexao.py*. Em seguida, é iniciado duas funções do *OpenCV*, a de captura dos vídeos e do detector de *QR Codes*.

A etapa seguinte trata da lógica de decodificação dos códigos detectados. Caso seja um código válido, é realizada a separação das 2 chaves, executando a função gravarRegistro, que recebe por parâmetro a chave codificada, demonstrado na Figura 28 a seguir.

Figura 28: Função gravarRegistro.

```
def gravarRegistro(s):
    if(getRegistro(s)==[]):
        if(getPart(s)!= null):
            idRegistro= getID()
            date = getHora()
            participante = getPart(s)
            data=(idRegistro, date, participante[0][0], participante[0][1], participante[0][2])
            print(data)
            sql ="INSERT INTO registro(idRegistro, dataRegistro, part_chave, part_idUser,"
            sql = sql + " part_idEvento) VALUES(%s, %s, %s, %s, %s)"
            mycursor.execute(sql,data)
            mydb.commit()
            return True
        else:
            date = getHora()
            data=(idRegistro, date, s, 0, 0)
            sql ="INSERT INTO registro(idRegistro, dataRegistro, part_chave, part_idUser,"
            ql = sql + "part_idEvento) VALUES(%s, %s, %s, %s, %s)"
            mycursor.execute(sql,data)
            mydb.commit()
            return False
    else:
        print("Registro ja existente!")
        return False
```

Fonte: Autor (2020).

Nesta função inicialmente é visto se já existe um registro com a chave recebida, caso tenha, irá exibir uma mensagem informando que o registro já foi realizado. Caso ainda não exista registro com essa chave, é verificado se esta chave está vinculada com o participante.

Caso não esteja vinculado a algum participante, é negado o acesso e registro a tentativa. Caso contrário, é liberado o acesso ao participante, e realizado o registro, gravando quem é o usuário, o evento que ele acessou, e a chave utilizada.

## 5.7 Testes e Resultados

Para a realização dos testes, visando avaliar a qualidade e o desempenho do protótipo, inicialmente foi feito o cadastro dos usuários, aos quais são atribuídas sua credencial básica, formada por um *login* e uma senha.

Na sequência, utilizando o perfil de administrador do sistema, foram cadastrados os eventos e, em seguida, vinculados os usuários que poderiam participar dele. Com isso, somente alguns usuários terão permissão de acesso ao local do evento, como é demonstrado na Figura 29, onde o usuário, está vinculado somente a um evento.

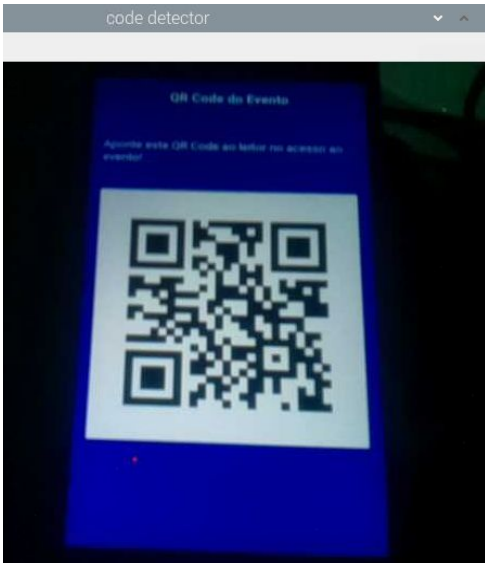
Figura 29: Lista de eventos do usuário.


Fonte: Autor (2020)

A partir destes vínculos, os usuários podem selecionar o evento no qual querem validar seu acesso, e assim visualizar o *QR Code* que é usado para validar este acesso. Com o *QR Code* exibido na tela, o usuário necessita apontá-lo para a câmera, que está acoplada ao *Raspberry Pi*.

Na sequência, o *QR Code* será capturado pela câmera e decodificado, e posteriormente será autenticado para validar este acesso. Como pode ser observado na Figura 30, a captura do *QR Code*.

Figura 30: Captura do *QR Code* pela *Raspberry Pi*.




Fonte: Autor (2020).

Por fim, ao validar o acesso, este registro será gravado no *log*. Permitindo ser consultado pelo administrador, a partir da tela de consultas de registros. A consulta de registros permite listar todos os registros gravados, tendo ou não o acesso liberado, como mostra a Figura 31.

Figura 31: Listagem dos registros.


Fonte: Autor (2020).

Nesta listagem é possível identificar os acessos liberados e negados, a chave utilizada e o momento em que esta tentativa de acesso ocorreu. Portanto, após a realização dos testes, a solução AuthToken superou as expectativas, visto que é um protótipo de baixo custo, e que apresenta um gerenciamento de acessos.

# 6 CONSIDERAÇÕES FINAIS

Durante o trabalho apresentou-se a solução AuthToken para a realização de autenticação de acesso a partir da utilização do *QR Codes*, uma tecnologia que vem crescendo recentemente, mas pouco explorada para a solução aqui desenvolvida. A solução ainda conta com consulta aos registros dos acessos, permitindo o monitoramento destes acessos por parte do administrador do sistema.

O método de autenticação utilizado foi baseado em propriedade, visto que o usuário precisa ter o *QR Code* em seu *smartphone* para poder realizar a autenticação do acesso. Esta autenticação ocorre a partir deste *QR Code*, que contém uma chave identificadora do usuário, única, que verifica se ele possui permissão para o acesso pretendido.

Após a realização dos testes, a solução se mostrou satisfatória no que se diz respeito à segurança e agilidade dos acessos. Esta é confirmada através dos testes realizados, onde a solução *AuthToken* correspondeu de forma adequada às tentativas de acessos, sejam elas válidas ou não, além de disponibilizar o monitoramento destes registros.

Portanto, esta solução pode ser aplicada em inúmeros casos, como acesso de eventos, laboratórios, bibliotecas, almoxarifados, entre outros, onde a solução possibilita ao administrador da aplicação gerenciar e monitorar os acessos destes locais.

Dessa forma, conclui-se que a solução *AuthToken* possibilita o gerenciamento e monitoramento de acessos com o uso de *QR Codes*, validando a

hipótese de que é possível desenvolver sistemas de controle de acesso funcionais e de baixo custo, a partir de dispositivos móveis.

## 6.1 Trabalhos Futuros

A partir dos resultados obtidos, pôde-se perceber que melhorias podem ser aplicadas, adicionando novas funcionalidades à solução. Tais como:

- Funcionamento *offline* do aplicativo e do leitor de *QR Code*.
- Implementação de um método mais robusto de autenticação como o *Firebase Auth* por exemplo que aumentaria a segurança.
- Sistema Web para a leitura dos QR Codes, substituindo o Raspberry Pi, para que possa ser utilizado em eventos online.
- Realizar testes de desempenho, medindo o tempo de resposta e a distância em que os QR Codes podem ser decodificados.
- Filtros adicionais para as telas de listagens para facilitar a pesquisa para os usuários, que podem variar conforme o mercado da aplicação do protótipo.

Portanto estas melhorias seriam muito úteis para serviços deste tipo, sendo então, indicadas para trabalhos futuros.

# REFERÊNCIAS


ANTUNES, Bruno. **Conheça os 3 tipos de métodos de autenticação**. Disponível em Blog Seguranca em Simples Atos: http://segurancaemsimples atos.com.br/blog/conheca-os-3-tipos-de-metodos-de-autenticacao. Acessado em 20 de maio de 2020.

BARROS, Aidil J. S.; LEHFELD, Neide A. S. **Fundamentos de Metodologia Científica.** 3ª Ed. Pearson Education. São Paulo, 2007.

BORGES, Luiz E. **Python para Desenvolvedores.** Ebook disponível em: https://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf. Acessado em 10 de junho de 2020.

BRAGA, Newton C. **Como funciona o *QR Code***. Disponível em Instituto NCB: https://www.newtoncbraga.com.br/index.php/como-funciona/15548-como-funciona-o-qr-code-art4052. Acessado em 25 de maio de 2020.

CERVO, Amado L. et al. **Metodologia científica.** 6ª Ed. Pearson Education. São Paulo, 2006.

COELHO, Maurício. ***QR Code*: o que é e como usar**. Disponível em portal IG : https://tecnologia.ig.com.br/dicas/2013-03-04/qr-code-o-que-e-e-como-usar. Acessado em 25 de maio de 2020.

DIMES, Troy. **JavaScript: Um Guia para Aprender a Linguagem de Programação JavaScript.** Editora Babelcube. 2015.

DONOHUE, Bryan. **O que é autenticação de dois fatores e como usá-la?**. Disponível em Blog Kaspersky Daily : https://www.kaspersky.com.br/blog/o-que-

e-a-autenticacao-de-dois-fatores-e-como-usa-la/3226. Acessado em 28 de maio de 2020.

FERNANDES, Rafael, et. al. **SAAS: uma Solução de Autenticação para Aplicativos de Smartphones.** Unipampa, Bagé, 2019.

FERREIRA, Débora C. ***APISIM* - Uma api restful para gerenciamento de recursos de sistemas operacionais.** Timóteo, 2017.

FREITAS, Andréia R. R. P. ***QR Code* - Tendência de Evolução Comercial no Ponto de Venda Físico de Retalho**. Universidade Europeia. Lisboa, Portugal, 2017.

International Organization for Standardization. **ISO/IEC 18004:2015. Information technology - Automatic identification and data capture techniques - QR Code bar code symbology specification**. Disponível em: https://www.iso.org. Acessado em 10 de junho de 2020.

JESUS, José H. **Estudo de tecnologias IOT para reconhecimento de imagens QR Code otimizado no controle de saída de produtos de almoxarifado**. Biblioteca da FEMA, Assis, 2018.

JOSÉ, Fábio R. S. **Ionic Framework Essencial.** Ebook disponível no Blog Fabio Rogerio SJ: https://fabiorogeriosj.com.br/2016/05/02/Ebook-1-Ionic-Framework-Essencial. Acessado em 10 de junho de 2020.

JUNIOR, Ronaldo C.M. **Modelo de Autenticação Multicanal Baseado em Proximidade**. PUC PR. Curitiba, 2018. Disponivel em: https://www.ppgia.pucpr.br/pt/arquivos/mestrado/dissertacoes/2018/Dissertacao_RonaldoCesarMengatoJunior.pdf. Acessado em 30 de maio de 2020.

KARASINSKI, Lucas. **O que significa cada quadro de um *QR Code***. Disponível em TecMundo : https://www.tecmundo.com.br/qr-code/37372-o-que-significa-cada-quadrado-de-um-qr-code. Acessado em 25 de maio de 2020.

LUZEMBO, Plamedi L. **Protótipo de Sistema de Automação Residencial Integrador com Raspberry Pi Utilizando Windows**. FURB. Blumenau, 2017.

MARCONDES, J. S. **Sistema de Controle e Acesso: O que é? Definições e como funciona.** Disponível em Blog Gestão de Segurança Privada: https://gestaodesegurancaprivada.com.br/sistema-controle-de-acesso-definicoes-como-funciona. Acessado em 21 de maio de 2020.

MEIRELLES, Fernando. **Brasil tem 230 Milhões de Smartphones em Uso**. Disponivel em Época Negócios do site Globo: https://epocanegocios.globo.com/Tecnologia/noticia/2019/04/brasil-tem-230-milhoes-de-smartphones-em-uso. Acessado em 21 de maio de 2020.

OLIVEIRA, Sergio. **Internet das Coisas com ESP8266, Arduino, Raspberry Pi**. Novatec Editora. 2017.

OTA, Fernando K.C. **Autenticação de Dispositivos Móveis usando NFC**. Repositório institucional UNESP. São José do Rio Preto, 2016.

PALMEIRA, Eduardo J.S.S; FERNANDES, Luiz R.A. **Controle de Acesso para Estádio de Futebol Utilizando Tecnologia NFC**. Repositório de outras coleções abertas ROCA. Curitiba, 2013.

REZENDE, Livia. ***QR Code*: o que é? e como usar**. Disponivel em Compara Plano: https://comparaplano.com.br/blog/qr-code. Acessado em 20 de maio de 2020.

SILVA, Gabriela C., et. al. **Transposição da Autenticação Federada para uma Solução de Controle de Acesso Físico no contexto da Internet das Coisas.** SBSEG, Porto Alegre, 2018.

SOARES, Elisa F. A. ***API* URB: Uma *api rest* para gerenciamento de problemas urbanos.** Mossoró, 2019.

XAVIER, Fox. ***QR Code*: entenda o que é e como funciona o código**. Disponivel em Tectudo: https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2011/03/um-pequeno-guia-sobre-o-qr-code-uso-e-funcionamento. Acessado em 20 de maio de 2020.